Req 5.4.1

Title:

Bias - doit

Objective:

Determine master bias frame.

The signal in raw scientific frames contains a component that is due to a bias current. This component shows up as an offset to the signal. The bias-offset has the following characteristics: i) the bias level grows to its asymptotic level in the first few hundred lines, and ii) the bias level depends on the total signal in a given line. Therefore, an initial bias correction—the **overscan correction**, is applied by averaging the overscan pixels for each line, and subtracting this value from that line. It is noted that the bias level may jump significantly after a cold restart of the FIERA.

In addition, the bias offset exhibits a residual pattern, which is measured by the master bias frame. To construct the master bias a series of 10 zero-second bias exposures is overscan-corrected, and then averaged, rejecting 5σ outliers (σ = dispersion of the 10 bias exposures of individual pixels), due to particle hits during read-out. The resulting master bias frames will be used for the correction of all frames. For each master bias frame the mean value for each CCD chip will be determined and evaluated in a trend analysis.

As the readout noise dominates the rms scatter in the bias frames, while the shotnoise of the sky background dominates the rms scatter on the sky images, which is nominally much larger than the readout noise, it is sufficient to characterize the bias value at individual pixels with an accuracy of (readout noise/$\sqrt{10}$).

A comparison with a previous master bias frame will be done as an evaluation of the overall health of the instrument and the quality of the data. This will thus measure short-term variations. Long term variations can be assessed using trend analysis.

A comparison of the mean level with laboratory values will be used as an overall quality check.

Fulfilling or fulfilled by:

Selfstanding. Raw data are also used for **req. 522** *Bad/hot pixels*

Calfiles used by: **req. 542** *Dome flat*, **req. 543** *Twilight flat* **req. 544** *Night sky flat* **seq. 632** *Trim, de-bias, flatfield*

When performed/frequency:

daytime- Commissioning, in RP initially daily. Later the frequency is to be determined by experience.

Sources, observations, instrument configurations:

10 observations with 0 (zero) seconds exposure time.

Inputs:

Raw data bias frames

CalFile– 541 *Master Bias frame* previous versions

Laboratory values of bias levels.

Outputs:

CalFile– 541 *Master Bias frame* to be used by **seq. 632** *de-bias flat field*

Required accuracy, constraints:

The required accuracy per pixel in the master bias frame is "nominal read-outnoise/$\sqrt{10}$".

For the quality check: Since an overscan correction is performed, the deviation of the mean level of master bias (bias level) from zero, should be less than TBD.

Estimated time needed:

Observation: 15 min. Reduction: $<$ 2 min./CCD.

Priority:

essential

TSF:

Mode– Stare N=10

TSF– OCAM_img_cal_bias

Recipe:

```
Bias -i <raw_bias_list> -r READNOISE [-oc OVERSCAN_CORRECTION]

raw_bias_list       : list of raw bias exposures
READNOISE           : readnoise value in ADU (float).
                      Range of allowed values: 1.0 - 5.0.
OVERSCAN_CORRECTION : overscan correction mode (integer).
                      Description of allowed values:
                        0: apply no overscan correction (default)
                        1: use median of the prescan in the
                           x-direction
```

```
                        2: use median of the overscan in the
                           x-direction
                        3: use median of the prescan in the
                           y-direction
                        4: use median of the overscan in the
                           y-direction
                        5: use the per-row value of the prescan
in
                           the x-direction
                        6: use the per-row value of the overscan
in
                           the x-direction
```

Before applying this recipe, use **Recipe– Split**—which is documented in **seq.–631**—with the `-t bias` option to split the raw multi-extension FITS input files.

Needed functionality:

image - processing (eclipse.trim_and_overscan());
image - arithmetic (eclipse.image_sub(), eclipse.image_mul(), eclipse.image_div(),

eclipse.image_add_local());
image - average (eclipse.cube_avg_median());
image - statistics (eclipse.iter_stat());
image - mask (eclipse.image_threshold2pixelmap(), eclipse.pixelmap_2_image())

CA:

Processing (make):
1. Trim the input raw bias frames, and do an overscan correction on the resulting images.
2. Combine the trimmed images into an average bias frame, using 5σ rejection.
3. Measure the statisics of the combined image.

Verification (verify):
1. Check that mean bias level is zero (less than TBD).
2. Check that the rms variance of the bias level is less than TBD.

Trend Analysis (compare):
1. Check that the difference of the rms with respect to previous version is less

than TBD.

CAP:

```
  MAXIMUM_ABS_MEAN        : Quality Control
  MAXIMUM_STDEV           : Quality Control

QC Flags:
  BIAS_MEAN_TOO_LARGE
  BIAS_STDEV_TOO_LARGE

Constants:
  SIGMA_CLIP          : clipping threshold for bad pixels in raw
                        input data (float). Value: 3.0.
  REJECTION_THRESHOLD : clipping threshold for bad pixels in the
                        resultant bias image (float). Value: 4.0.
  MAXIMUM_ITERATIONS  : number of iterations in bias statistics
                        measurement (integer). Value: 3.
```

```
reduced_bias_list = []
for raw_bias in raw_bias_list:
    reduced_bias = eclipse.trim_and_overscan(raw_bias)
    reduced_bias_list.append(reduced_bias)

# first estimate of mean
median = eclipse.cube_median(reduced_bias_list)

# Find outliers in first reduced bias image
dev = eclipse.image_sub(median, reduced_bias_list[0])
good = eclipse.threshold2pixelmap(dev, -SIGMA_CLIP * READNOISE,
                                  SIGMA_CLIP * READNOISE)
good = eclipse.pixelmap_2_image(good)

# record these values
sum_data   = eclipse.image_mul(reduced_bias_list[0], good)
sum_pixels = good

# Find outliers in remaining images, and reject these too
for reduced_bias in reduced_bias_list[1:]:
```

```
    dev = eclipse.image_sub(median, reduced_bias)
    good = eclipse.threshold2pixelmap(dev, -SIGMA_CLIP * READNOISE,
                                      SIGMA_CLIP * READNOISE)
    good = eclipse.pixelmap_2_image(good)
    eclipse.image_add_local(sum_data,
                            eclipse.image_mul(reduced_bias, good))
    eclipse.image_add_local(sum_pixels, good)

# new mean
bias_image = eclipse.image_div(sum_data, sum_pixels)
bias_statistics = eclipse.iter_stat(bias_image, MAXIMUM_ITERATIONS,
                                    REJECTION_THRESHOLD)

# Quality Control
if abs(bias_statisics.avg_pix) > MAXIMUM_BIAS_MEAN:
    BIAS_MEAN_TOO_LARGE = 1
if bias_statisics.stdev > MAXIMUM_BIAS_RMS:
    BIAS_RMS_TOO_LARGE = 1
diff = bias_statisics.stdev-previous.bias_statistics.stdev
if abs(diff) > MAXIMUM_BIAS_RMS_DIFFERENCE:
    BIAS_RMS_DIFFERENCE_TOO_LARGE = 1
```